Arnold Willemer

Coding for Fun

mit C++

- Das nächste Level für C++-Programmierer!
- Spannende Programme verstehen und selbst entwickeln
- Nachtfahrt, Frosch & Eichhörnchen, Eliza, Labyrinth, Navis, Mondlandung u.v.m.

Auf CD: C++-Compiler, Tools und natürlich alle Code-Beispiele des Buchs

Galileo Computing

Coding for Fun mit C++
Willemer
1512

Auf einen Blick

Inhaltsverzeichnis

Programmieren macht Spaß.

1 C++ – Das Porträt

Sie programmieren in C++. Dann sind Sie hier richtig. Sollten Sie diese Sprache dagegen noch nicht beherrschen, will ich nicht ausschließen, dass Sie dieses Buch an der einen oder anderen Stelle etwas unverständlich finden werden. Aber da gibt es Abhilfe. Im gleichen Verlag finden Sie ein gutes Buch namens »Einstieg in C++«. Die Namensgleichheit des Autors ist nicht einmal zufällig. Vielleicht lesen Sie zunächst jenes Buch und dann dieses.

Die Anfänger dürften nun das Buch zugeklappt haben und erst in ein paar Tagen oder Wochen wieder zu uns stoßen, nachdem sie C++ gelernt haben.

Sie als Kenner der Sprache C++ werden in diesem Kapitel die Gelegenheit haben, etwas über die Geschichte und den Charakter von C++ zu erfahren. Sie werden also all das erfahren, was Sie noch nie hören wollten oder längst schon wissen. Daneben werden Sie lesen, wie dieses Buch konzipiert ist. Falls Sie das Buch aber bereits gekauft und bezahlt haben oder wenn Sie lieber überrascht werden wollen, schlage ich vor, Sie überblättern dieses Kapitel und sparen damit Zeit.

Spätestens jetzt müsste ich allein sein und sollte mir die Frage stellen, warum ich dieses Kapitel geschrieben habe, wenn es doch keiner liest.

1.1 Buchkonzept

Programmieren ist eine besondere Form des Knobelns. Wer programmiert, löst täglich Denksportaufgaben. Der eine oder andere hat diese Knobelei zu seinem Beruf gemacht. Aber die typischen Aufgaben im Beruf sind meist eher technischer Natur. Und so wie die nichtprogrammierenden Artgenossen das Sudoku in der Tageszeitung lösen, hat der Programmierer hin und wieder Lust, etwas zu programmieren, was einfach nur Spaß macht.

In diesem Buch sind Themen aufgegriffen worden, die sich für solche Knobeleien eignen. Die Kapitel bauen nicht aufeinander auf, sondern können weitgehend in beliebiger Reihenfolge gelesen werden. Sie sind nicht völlig ohne Sinn und Verstand in dieser Reihenfolge angeordnet worden, aber beinahe.

In jedem Kapitel wird zunächst in das Thema eingeführt. Dann wird ein Lösungsansatz vorgestellt und der Quelltext beschrieben. Schließlich werden Ihnen ein paar Ideen aufgezeigt, wie Sie die Programme erweitern können. Ich kann mir vorstellen, dass Sie selbst eigene Ideen haben.

Ich verstehe dieses Buch durchaus als Mitmachbuch. Warum sollte ich den Spaß auch allein gehabt haben. Der Verlag stellt freundlicherweise zu diesem Buch ein Forum bereit: *http://www.galileocomputing.de/forum/gp/forumID-241*. Stellen Sie dort Ihre Lösungen vor. Es wäre nett, von Ihnen zu hören.

Die vorgestellten Programme sind also nicht komplett und fertig, sondern eher Appetithappen. Sie erheben auch nicht den Anspruch, vorbildliches C++ zu predigen. Und darum werden Sie vielleicht an der einen oder anderen Stelle bemängeln, dass die Fehlerbehandlung fehlt, Klassen mit öffentlichen Membervariablen definiert sind und andere Verbrechen gegen die Ideologie der sauberen Programmentwicklung begangen wurden. Der Grund dafür ist, dass die Programme möglichst kurz und übersichtlich sein sollten. Sie sollen Spaß machen, nicht Vorbild sein.

1.2 Blick zurück

Wer die Geschichte der Programmiersprache C++ betrachten möchte, sollte ein Fernglas verwenden, schließlich geht es weit zurück in die Vorzeit. Denn damals, kurz vor der Erfindung der Rasierklinge, entwickelten Brian Kernighan und Dennis Ritchie die Programmiersprache C, um das Betriebssystem UNIX nicht in Assembler schreiben zu müssen.

Eine unbekannte Quelle (Bernhard L. Hayes, NetNews-Gruppe) schrieb in einem Artikel unter dem Titel »Erfinder von UNIX und C geben zu: alles Quatsch!«, dass UNIX und C nur ein Aprilscherz gewesen sei.

> *In einer Ankündigung, die die Computerindustrie verblüffte, haben Ken Thompson, Dennis Ritchie und Brian Kernigham zugegeben, dass das Betriebssystem Unix und die Programmiersprache C ein raffinierter Aprilscherz sind, der sich über 20 Jahre am Leben erhalten hat.*
>
> *Bei einem Vortrag vor dem letzten UnixWorld-Software-Entwicklungsforum enthüllte Thompson:*
>
> *1969 hatte AT&T gerade die Arbeit am GE/Honeywell/AT&T-Multics-Projekt beendet. Brian und ich experimentierten zu dem Zeitpunkt mit einer frühen Pascal-Version von Professor Niklaus Wirth vom ETH-Laboratorium in der Schweiz und waren beeindruckt von seiner Einfachheit und Mächtigkeit. Dennis hatte »Der Herr der*

Klinge« gelesen, eine spöttische Parodie auf Tolkiens große Triologie »Der Herr der Ringe«.

Im Übermut beschlossen wir, Parodien zur Multics-Umgebung und zu Pascal zu verfassen. Dennis und ich waren für die Betriebssystemumgebung verantwortlich. Wir sahen uns Multics an und entwarfen ein neues System, das so komplex und kryptisch wie möglich sein sollte, um die Frustration der gelegentlichen Nutzer zu maximieren. Wir nannten es Unix, in Anspielung auf Multics, und fanden es auch nicht gewagter als andere Verballhornungen. Danach entwickelten Dennis und Brian eine wirklich »perverse« Pascal-Version namens »A«. Als wir bemerkten, dass einige Leute tatsächlich versuchten, in A zu programmieren, fügten wir schnell einige zusätzliche Fallstricke hinzu und nannten es B, BCPL und schließlich C. Wir hörten damit auf, als wir eine saubere Übersetzung der folgenden Konstruktion erhielten:

```
for(;P("\n"),R--;P("!"))for(e=C;e--;P("_"+(*u++/8)%2))
```

Der Gedanke, dass moderne Programmierer eine Sprache benutzen würden, die solch eine Anweisung zuließ, lag jenseits unseres Vorstellungsvermögen. Wir dachten allerdings daran, alles den Sowjets zu verkaufen, um ihren Computerfortschritt 20 Jahre und mehr zu behindern. Unsere Überraschung war groß, als dann AT&T und andere US-Unternehmen tatsächlich begannen, Unix und C zu verwenden! Sie haben 20 weitere Jahre gebraucht, genügend Erfahrungen zu sammeln, um einige bedeutungslose Programme in C zu entwickeln und das mit einer Parodie auf die Technik der 60er-Jahre! Dennoch sind wir beeindruckt von der Hartnäckigkeit (falls nicht doch Gemeinsinn) des gewöhnlichen Unix- und C-Anwenders. Jedenfalls haben Brian, Dennis und ich in den letzten Jahren nur in Pascal auf einem Apple Macintosh programmiert, und wir fühlen uns echt schuldig an dem Chaos, der Verwirrung und dem wirklich schlechten Programmierstil, der von unserem verrückten Einfall vor so langer Zeit ausging.

Namenhafte Unix- und C-Anbieter und -Benutzer, einschließlich AT&T, Microsoft, Hewlett-Packard, GTE, NCR und DEC haben vorläufig jede Stellungnahme abgelehnt. Borland International meinte, sie hätten diesen Verdacht schon seit Jahren gehegt und würden nun dazu übergehen, ihre Pascal-Produkte zu verbessern und weitere Bemühungen um die C-Entwicklung stoppen. Ein IBM-Sprecher brach in unkontrolliertes Gelächter aus.

Den Artikel finden Sie in verschiedenen Varianten unter den Stichworten »UNIX«, »Quatsch« und »Aprilscherz« über jede gut sortierte Suchmaschine mehrfach im Internet zitiert, beispielsweise auch unter folgender URL:

http://www.c-plusplus.de/geschichte.htm

Taufe

Die Namensgebung von C ist etwas ungewöhnlich, und so spinnen sich Legenden um deren Entstehung. Eine unbestätigte Legende besagt, dass zuerst der Name A im Gespräch war. Aber die Paten fürchteten, dass das A vielleicht in einigen Jahren von Apple patentiert werden würde. Und so ging man das Alphabeth durch. B kam nicht in Frage, weil es mit B-Ware oder gar mit BASIC in Verbindung gebracht werden könnte. C hatte dagegen eine freundliche Assoziation. Man könnte Zitronen wählen, um die Lehrbücher über die neue Programmiersprache zu schmücken, und das Gelb würde einfach gut aussehen.

Der Haken an dieser Legende ist, dass die Firma Apple erst ein Jahrzehnt später auf den Plan trat. Außerdem hätten die Namensgeber dann übersehen, dass die Sprache COBOL mit C anfängt. Und das ist sicher eine noch schlimmere Assoziation als BASIC.

Eine deutlich häufiger wiederholte Geschichte besagt, dass man die Sprache erst A nannte, eine spätere Entwicklungsstufe B und dann zu C kam. Und dann hätten sie einfach keine Lust mehr gehabt, die Sprache schon wieder zu verändern. Es war auch ein guter Moment zu stoppen. Die nachfolgenden Buchstaben wären durch D-Zug, E-Mail, F-Wort, G-Punkt und H-Spalterei belegt gewesen. Und wer hätte schon eine Programmiersprache namens I ernst genommen?

Bjarne Stroustrup wusste wohl um die Problematik, sodass er seine Erweiterung der Sprache C um Klassen eben C++ nannte, also ein Inkrement der Sprache C.

Tempo!

Ganz egal wie es nun zu dem Namen kam. Die Qualitäten der Sprache waren durch das Umfeld gezeichnet. Bis dahin wurden Betriebssysteme in Assembler geschrieben. Dafür gab es vor allem drei Gründe. Assemblerprogramme waren klein, schnell und hatten leichten Zugriff auf die Hardware. Assembler hatte aber auch den Nachteil, dass es das entstehende Betriebssystem zwingend mit einem Prozessor verheiratete. Weitere Nachteile lagen darin, dass die Programme lang und unübersichtlich waren und später kaum wartbar waren. Als UNIX entwickelt wurde, sollte es von der Hardware unabhängig sein und dennoch der Geschwindigkeit von Assembler möglichst nahe kommen.

Maschinennah

Der Inkrementoperator ist ein Beispiel für diese Optimierung. Fast jeder Prozessor hat einen Inkrementoperator, der schneller ist als die Addition mit 1. Dieser konnte von C direkt angesprochen werden.

Die Zeiger waren von Anfang an so ausgelegt, dass ein direkter Zugriff auf die Controller-Bausteine möglich ist. Controller-Bausteine haben eine feste Speicherstelle

im Adressraum. Weist man diese Adresse der Zeigervariablen direkt zu, kann anschließend der Inhalt des Controllers direkt gelesen und geschrieben werden, und schon war die Verwendung von Assembler zu diesem Zweck unnötig.

C bietet also die Möglichkeit, einen Computer systemnah zu programmieren. Der Zugriff auf Zeiger, das Anfordern und Freigeben von Speicher und der Umgang mit systemnahen Komponenten erfordert eine gewisse Disziplin. Wie in vielen anderen Lebensbereichen ist es auch hier gut, wenn man weiß, was man tut. Die Idee, dass eine Programmiersprache den Computer vor dem Programmierer beschützt, ist jedenfalls in C nicht implementiert.

Moderne Sprachkonzepte

C war aber mehr als ein portabler Assembler. Die damals mit ALGOL aufgekommene strukturierte Programmierung wurde bereits unterstützt. Es gab Schleifenbefehle, Funktionen mit Parametern und lokale Variablen. Darüber hinaus konnten Datenstrukturen modelliert werden, was in jenen Tagen nicht selbstverständlich war.

In jenen Tagen gab es einen erbitterten Kleinkrieg zwischen den Verfechtern der klassischen Programmiersprachen wie COBOL, FORTRAN und BASIC und der aufkommenden strukturierten Programmierung. Die alten Programmiersprachen verwendeten ausgiebig den Befehl GOTO, was dazu führte, dass es sehr schwierig war, dem Source-Code zu folgen, weil die logischen Stränge so durcheinander waren wie die Spaghettis auf einem Teller.

Die Sprache C stammte ähnlich wie PASCAL aus der ALGOL-Linie, die Mechanismen bot, die diesen Spaghetti-Code vermeiden konnte. Allerdings gibt es Programmierer, die mit jeder Sprache in FORTRAN programmieren können. Ein etwas umfangreicherer Artikel, der leicht ironisch die damalige Diskussion widerspiegelt heißt »Echte Programmierer meiden Pascal« und ist vollständig unter folgender URL zu finden:

http://www.leo.org/information/freizeit/fun/pascal.html

Wer es schätzt, Dinge in der Originalsprache zu lesen, findet eine englische Version unter der folgenden URL:

http://www.crusoe.de/pascal.htm

Interessanterweise gab es zu jener Zeit auch heftigen Streit zwischen den C- und den PASCAL-Programmierern, welche Sprache die beste sei. Oft fanden sich die Wortgefechte in den einschlägigen Newsgroups. So manch einer Diskussion merkte man an, dass Programmierer wohl nicht so oft wie andere Menschen ihre diesbezüglichen animalischen Veranlagungen zu Stadiengesängen bei der Bundesliga auslebten.

Index

M

N

O

P

R

S

T

U

V

W

X